Epsilon: Patching Mobile WiFi Networks

Hamed Soroush, Nilanjan Banerjee, Mark D. Corner, Brian Neil Levine
Dept. of Computer Science, University of Massachusetts, Amherst
{hamed, nilanb, mcorner, brian}@cs.umass.edu

ABSTRACT

Open Wi-Fi networks offer a chance to have ubiquitous, mobile connectivity by opportunistically leveraging previously deployed resources. Open Wi-Fi access points are densely deployed in many cities, offering high bandwidth at no cost to the mobile node. Unfortunately, Wi-Fi networks are riddled with coverage holes, resulting in poor network performance, even if planned for blanket coverage. To back this claim, we present the results of a measurement study of a small city's Wi-Fi network—both planned and unplanned—using mobile nodes, verified with data collected from a second city. We find that holes can be broadly classified into two categories: (1) *permanent* holes due to a lack of Wi-Fi coverage; and (2) *transient* holes that are due to mobility and channel characteristics. We show that these holes have a severe, adverse effect on the performance of network transport protocols.

Unfortunately, fixing these holes by adding WiFi base stations is an expensive and difficult process–there is not always the connectivity, power, and legal authority, to place equipment. Instead, by enhancing the network with a broader area, but still unlicensed, backbone channel we can patch holes in connectivity. This broad area network is low-bandwidth, but as we show in this paper, the backbone radio has a *multiplicative* effect on bandwidth because it keeps the mobile user's TCP's congestion window open and preventing retransmission timeouts on the high-bandwidth Wi-Fi channel. This effect comes with no modifications to the TCP protocol or stack, making it a generally deployable solution. Moreover, the low bandwidth radio has low energy consumption, allowing us to cover holes with solar-powered devices. We evaluate the effectiveness of this system, named Epsilon, for improving the performance of TCP/UDP sessions for a wide range of application workloads. We show that Epsilon produces a $2x$ to $13x$ improvement in TCP throughput while providing nearly ubiquitous connectivity at low cost.

1. INTRODUCTION

There are several methods of providing reliable, ubiquitous connectivity for mobile devices. Cellular deployments, including 3G, EVDO, and GPRS, offer commercial, fee-based coverage of large areas. Unfortunately, such infrastructure is costly and difficult to manage, and its installation and operation is reserved for a handful of large carriers. And due to the costs involved — a recurring fee of about US$50 per month per device — its use is limited to persons that can afford it, and usually for only one mobile device.

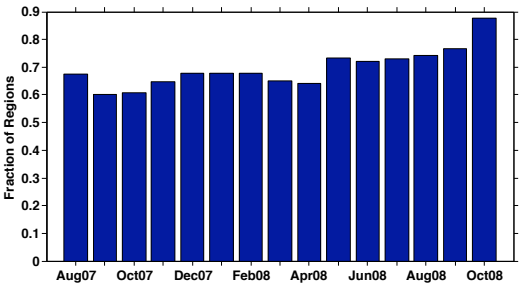

Figure 1: The fraction of 100x100 m^2 regions in our city where vehicles have open Wi-Fi access points.

At the same time, open WiFi access points (APs) are gaining in popularity [24] and are present in cities large [9,24] and small [6]. The major advantage of these organically deployed APs is cost — while 3G price plans will vary, open WiFi is by definition free to mobile users. Accordingly, open WiFi removes a significant impediment to pervasive computing. Figure 1 shows the availability of open WiFi APs in a section of our city. From Aug 07–Oct 08, at least 75% of 62500 regions each 0.01 km^2, supported open WiFi Internet access.

The disadvantage of WiFi access is robustness. Although WiFi links can have higher peak downstream bandwidths than 3G, it is a shorter-range radio, which leads to both coverage holes and areas of high loss rates, even in networks planned for blanket coverage [4, 18]. In networks where mobile users are subject to longer periods without connectivity, a myriad of ad hoc and disruption tolerant networking (DTN) protocols have been proposed that leverage the mobile nodes to deliver data [10]. While ad doc networks and DTNs provide connectivity where there was none, delivery delays depend on the mobility of other users. Typically, popular interactive, delay-sensitive applications cannot be reliably supported.

In this paper, we propose to enhance existing deployments of WiFi networks by adding small amounts of infrastructure. Specifically, we propose a system called *Epsilon*, which uses a novel approach of placing very low-bandwidth, long-range, radios wherever holes are present. The low-rate backbone acts as a bridge to an 802.11 AP. Connections from the mobile user are striped across both channels to smooth hand-off. Combined with an Internet proxy, clients can then hold TCP

connections across open APs, making applications more predictable. Because the radios are longer range than WiFi, fewer devices and a smaller cost is required to cover a large area.

Our experiments show, perhaps counter-intuitively, that the second radio has a *multiplicative* effect on the overall bandwidth: a 115 kpbs backbone covering 802.11 holes tens of seconds long can increase the aggregate TCP throughput of the mobile devices by several hundred percent. This paradox is explained by the ability of the low-bandwidth channel to keep the TCP sender's window large despite connectivity problems, enabling quick recovery when a WiFi connection returns. Further, this enhancement works with no modifications to TCP, making it easy to deploy in existing systems.

How much does network performance increase for mobile users of unplanned WiFi networks enhanced with inexpensive, low-bandwidth infrastructure? To quantitatively answer this question, we first analyze the prevalence of coverage holes in our outdoor WiFi testbed and one outdoor testbed in another city. Our experiments show that both permanent and transient holes are rampant in WiFi mobile network with disruption lengths of 5 seconds to 15 minutes. Then we quantify how TCP performance is improved by filling holes with the low-bandwidth bridge. Our results are based on data transfers over an operational prototype and a workload based on traces of WiFi connectivity by mobile nodes. We show that while existing solutions [9] can increase TCP throughput by a factor of 2x when a node is associated to an access point (effectively a gain of 15% when there are disruptions), using Epsilon, TCP throughput can increase for mobile users by 1.2x to 13.0x.

2. CONNECTIVITY MEASUREMENT STUDY

To motivate the need for Epsilon, we have conducted a measurement study of a wide-area, municipal Wi-Fi network, including a few dozen planned APs, and hundreds of organic, open WiFi networks. We have also validated the results of this study against traces from a network in a second city. Our results show that coverage holes are rampant in both environments.

Previous work has studied disruptions, both short term [5] and long-term [21]. Our study has three significant differences. First, we study link layer and application layer performance together, while previous work handled these layers in isolation [18]. Second, we measure coverage holes in both managed and unplanned Wi-Fi deployments, whereas most previous work concentrates on self-deployed, managed APs [18]. Finally, our analysis defines holes as a period of time when we can not hear beacons from any access point; in other words, we explore the effects of using one of many multi-AP technologies, including FatVAP [15], Juggler [20], or ViFi [5]. Most previous work calculated connectivity in terms of a single AP.

However, studying coverage using beacons alone embodies an opportunistic view of connectivity—APs may not really provide open access as many use web-based passwords of MAC address access control. Thus we can also measure holes using a more conservative measure: connect to APs and attempt to send data to a known host on the Internet. The holes in the network are bounded by these two kinds of measurements, one pessimistic and one optimistic.

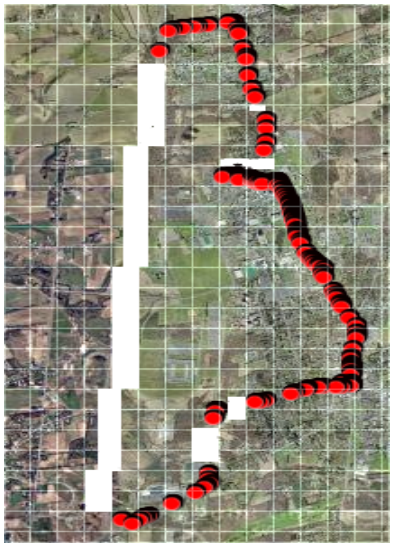

Figure 2: The map of the route followed by our transit cars. The red dots depict the locations of open access points from `kismet`. The white regions show the areas without any Wi-Fi connectivity.

Unfortunately, found that none of the available software which provide for simultaneous selection of multiple access points, could either be used over our equipment and operating system, had not been publicly released, or was not robust. Thus, our measurements of real connectivity is only limited to one AP at a time.

2.1 Measurement Methodology

Our measurements and evaluations are based on experiments we performed using more than 30 vehicles in a city with planned Wi-Fi Internet access points (APs) as well as third-party open Wi-Fi. The vehicles carry a Linux system (2.6.22.14 kernel), Atheros AR413 Wi-Fi card (with a 3 dBi antenna and the MadWiFi driver), and a GPS unit based on the SIRF Star III chipset. We have collected two measurements sets:

Measurement Set I: We configured the vehicles to associate with available Wi-Fi APs and immediately test end-to-end connectivity using `ICMP ping` to an known Internet host. For a fraction of all contacts, the vehicles initiate TCP sessions with the same host. The result of these trials is a log of each vehicle-to-AP contact in terms of duration, locations, vehicular speed, and the amount of data transferred. The vehicles select APs for association based on highest received signal strength indication. APs that are open but do not offer end-to-end connectivity are blacklisted for efficiency. The vehicles also log association failure events such as failure to

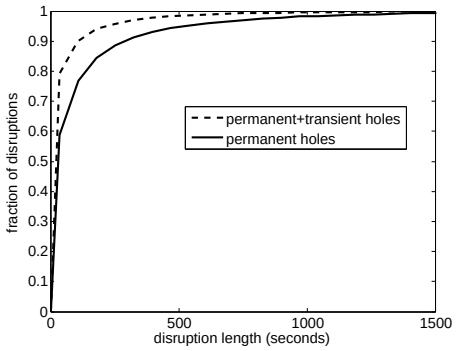

Figure 3: The a cumulative distribution function of the disruption lengths in our testbed. The data is based on associations with open access points from transit vehicles over a period of a month.

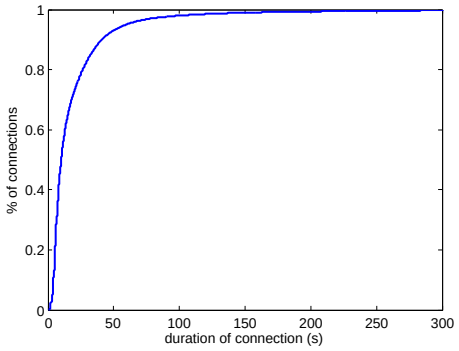

Figure 4: The cumulative distribution function of the duration of time mobile nodes in our network could remain connected to the Internet using open Wi-Fi access points.

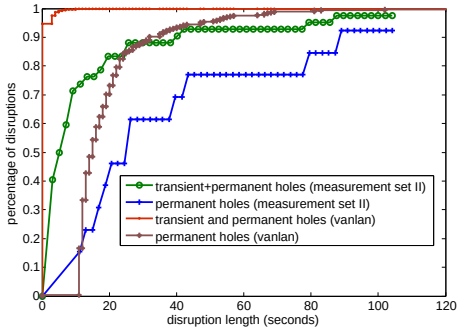

Figure 5: The cumulative distribution function of the disruption lengths calculated from link layer Wi-Fi beacons from open access points. The disruption periods correspond to times when no beacons were heard by the mobile node. Both large (greater than 10 seconds) and small disruption (less than 2 seconds) occur in both VanLan data-set and our collected traces.

obtain DHCP leases. In sum, our data represents approximately 9500 driving hours in the month of February 2009. During this period the vehicles saw 10056 unique open Wi-Fi access points. Given that the measurement connects to one AP at a time, and sometimes incurs delay in client-driven hand-off, this is a pessimistic view of connectivity.

Measurement Set II: For a more focused set of measurements, we used two vehicles traversing a shorter route, shown in Figure 2. Unlike the DHCP/ping experiments, the beacon measurements circumvent the impact of AP selection mechanism and provide for analysis of the effects of transient factors, including interference, mobility, and channel characteristics. The route includes residential and downtown areas of dense AP coverage, as well as areas with relatively sparse coverage. We used `kismet` to collect GPS-stamped 802.11 beacons from open Wi-Fi access points. These link layer beacons include the timestamp, received signal strength, channel noise, BSSID of the access points, and authentication information. The vehicles logged about 10 hours of data on five different days in 2009.

To generalize our results to other cities, we repeat our beacon analysis on the VanLAN data set [5]. The publicly available data set contains timestamped 802.11 beacons measured by a van over a period of five days from over 800 access points. Though the data does not reveal any information on whether the APs are secured, we optimistically assume that all the APs are open. This is consistent with using beacons as an optimistic measure of connectivity.

2.2 Coverage Holes

A myriad of factors can cause coverage holes, including mobility, 802.11 channel characteristics, association failures, and lack of Wi-Fi coverage. Using our measurement data, we detail the significance of each of these factors. We separate coverage holes into two broad categories. *Permanent holes* are periods of time when the mobile node is outside radio range of any open access point. *Transient holes* are periods of time when a node is within range of one or more access points but does not receive beacons from any APs due to packet loss or other problems.

Coverage holes can be identified by the time between consecutive successful associations from a mobile node. The duration of coverage holes is dependent on vehicle speed as well as environmental factors, such as shadowing. Figure 3 shows the distribution of the length of the holes in our mobile network based on Measurement Set I.

Permanent Holes.

Both planned and unplanned Wi-Fi networks can have areas where permanent coverage holes are present. Figure 2 shows a map of our network overlayed with available (open) Wi-Fi access points observed in Measurement II. In white, the map shows large areas of the network where there is no AP coverage.

The distribution of the length of disruptions, as shown as the solid curve in Figure 3, is highly skewed with a few long disruptions and a large number of smaller disruptions. The absolute lengths of the holes is high: the median length is 50

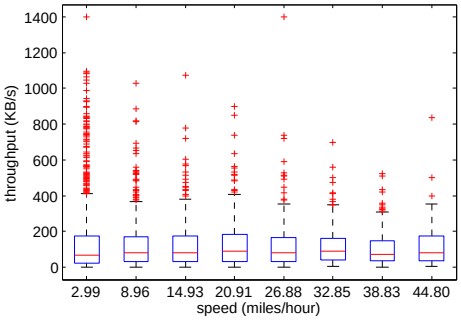

Figure 6: A box plot of the vehicle speed and TCP throughput observed at the vehicles of our testbed. The small correlation between the two demonstrates that mobility is not a primary cause of the transient holes in our network.

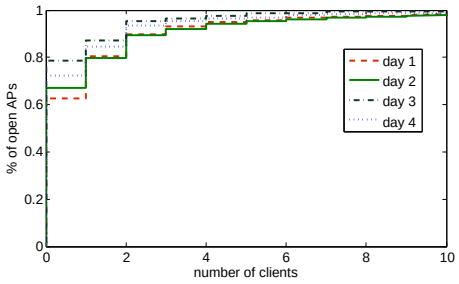

Figure 7: Number of wireless clients using open access points in our network at a particular instance of time. Small number of clients rules out AP congestion as a primary cause of transient holes.

seconds with a 90^{th} percentile of 500 seconds. In order to filter out the overhead of transient association failures (such as getting a `dhcp` lease) in identification of the holes, we have calculated the distribution of the time between two consecutive associations events (successful or not) and depicted it as the dashed curve in Figure 3. The result shows that while a fraction of holes are due to transient association failures, a large fraction of them occur because vehicles do not see any open access points.

To eliminate the effect of association overheads (such as getting a `dhcp` lease), we consider the solid line in Figure 3. For this experiment, we define a hole as the time between two association events (successful or otherwise). This filters out the overhead of transient association failures. While a fraction of holes are due to transient association failures, a large fraction of holes occur because vehicles did not see any open access points. We compare the distribution of lengths of time with and without connectivity in Figure 3 and Figure 4. The median connection and disruption lengths (for permanent holes) are comparable (median of 20 seconds). In the next section, we show that such disruptions can have an adverse effect on the performance of TCP and UDP.

In Measurement II, the median duration of the coverage holes was 10 seconds, though the 90^{th} percentile is 150 seconds. This skewed distribution is due to areas of dense AP coverage and other areas with no AP coverage, characteristic of unplanned Wi-Fi deployments. Using a similar technique to analyze the VanLAN data set reveals a similar distribution of permanent coverage holes, as shown in Figure 5.

Transient Holes.

While permanent holes occur primarily due to lack of Wi-Fi coverage, other factors such as channel interference, congestion, and mobility could lead to transient holes even when the mobile node is associated and transferring data using an open access point. To study these transient effects, we look at effect of mobility and channel characteristics.

Figure 6 shows the correlation between TCP throughput and the speed of the vehicle. The result shows that variations in speed does not have a significant effect on the amount of throughput obtained by the mobile vehicles. Hence, while mobility can cause transient holes during connection events, remaining likely causes of these holes are channel characteristics (interference)or possibly congestion at the access points.

To examine other factors creating transient holes, we analyze the 802.11 beacons collected in Measurement Set II. Figure 5 shows the distribution of holes greater than two seconds. Comparing to the distribution of permanent holes, we see that a large fraction of the holes are small. Given that every AP transmits beacons at the rate of once every 100ms, a two second interval would correspond to a period of time when at least 20 beacons are lost in succession (assuming only one AP within range). This interval reliably detects holes in coverage. Comparing the distributions in Figure 5, we find that a fraction of the disruptions are small (order of 2-3 seconds) and are likely due to transient effects. A similar result can be seen in the same figure for the VanLAN data set.

Figure 7 shows the distribution of the number of unique clients actively sending data to an access point during our experiments. The data was collected by sniffing packets to and from an access point as part of our Measurement Set II. From the figure we see that the median number of clients transferring data with the open access points is small, hence congestion is likely not the cause of transient holes. Therefore, we attribute the cause of transient holes to channel characteristics such as interference from other sources in the unlicensed band, and physical obstructions.

3. PERFORMANCE IMPACT

TCP and UDP streams suffer differently from disruptions in connectivity. Because of TCP's congestion control mechanisms, a mild disruption can engage slow start, strangling throughput. UDP simply suffers the minimum of the available capacity and offered packet rate. In this section, we *(i)* quantify the loss of throughput for TCP based on traces of mobility and coverage holes in our environment; and *(ii)* we present a model of TCP disruption that allows us to analytically determine the effect from coverage holes and provides

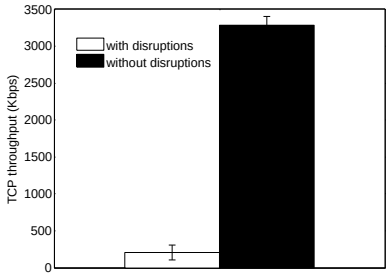

Figure 8: The decrease in TCP throughput for four minute TCP sessions in Measurement Set I. For this figure the disruptions correspond to times in between associations (successful or otherwise)

insight into how to use a background channel to improve throughput. Our results show that a small amount of background bandwidth is sufficient to overcome timeouts, keeping the TCP window open for the next Wi-Fi connection.

In some ways, disruptions are similar to random packet losses that cause similar TCP over-reactions [3]. However, when compared to common RTTs, the relative length of disruptions due to coverage holes and "disruptions" due to random packet losses, have very different effects—coverage holes have a more deleterious effect on throughput and require different mechanisms to solve.

3.1 Performance Study

To isolate the effect of coverage holes on TCP and UDP, we performed a series of trace-driven experiments based on Measurement Set I. To provide repeatable experimentation we conducted the experiments in an indoor environment with a stationary node connecting to a WiFi AP. We implemented a proxy at the client that uses `ip_queue` to drop packets whenever the trace recorded a disruption. We then started a large TCP transfer to download data from a host on the network. In these experiments, we assume that the outdoor WiFi network supports hand-offs, and a fixed IP address, negating the need to reestablish the TCP connection, even in the face of disruptions; we revisit connection establishment in later sections.

The results of the experiment are shown in Figure 8. As a point of reference, we plot the bars corresponding to TCP performance when there are not disruptions. We see that due to constant TCP timeouts and congestion control management TCP throughput decreases by a factor of 16 in the presence of disruptions, even though the disruptions would ideally decrease throughput by a factor of 2.

These empirical results are fully dependent on the mobility and connectivity patterns of our vehicles. In a second set of experiments, we isolated the dependence on the connection and disconnection times on TCP throughput. We use the same experimental set up as above, except that we strictly control the periods of connection and disconnection.

Figure 9 shows that when the connection is never disrupted, TCP throughput is about 2,600 Kbps. In this experiment, we

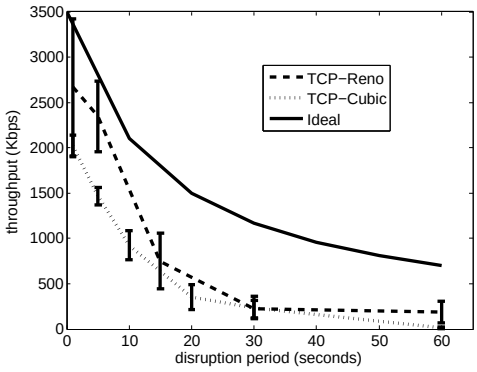

Figure 9: The figure shows the throughput for different flavors of TCP as a function of the disruption period. The connected period is chosen uniformly at random between 0-30 seconds. The ideal line corresponds to the best case TCP throughput achievable given the disruption lengths.

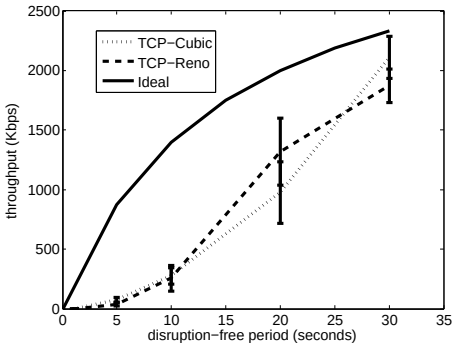

Figure 10: The throughput for different flavors of TCP as a function of the connected period. The disruption period is chosen uniformly at random between 0-30 seconds.

increase the period of disruptions from 0 to 60 seconds. We chose the length of connections uniformly at random from 0–30 seconds long. In this scenario, TCP performance falls sharply; for example, when disruptions are 30 seconds long, TCP throughput falls by a factor of 12. Compare this factor with the ideal decrease of 3, i.e., proportional to the periods of connectivity. Note that the figure shows the throughput of TCP Reno and TCP Cubic. While TCP Cubic is the default TCP implementation in the 2.26 Linux kernel, TCP-Reno is the most popular version of TCP. Figure 10 shows the dual experiment: the connection period is the independent variable, going from 0 to 30 seconds on the x-axis. The length of disruption is chosen uniformly at random from 0–30 seconds. Again we see that disruptions decrease throughput disproportionately to the "ideal" curve that is strictly based on the proportion of connectivity to disconnectivity.

3.2 A Model of TCP Disruption

The drop in TCP throughput is due to TCP's inability to adjust the proper timeout values and RTT after a connection

returns. In this section, we derive t_w: the amount of time that TCP stalls due to a disruption of length T_d. Our analytical framework is a model of TCP-Reno. We extend the well-known model defined by Padhye et al. [22] to include the effects of disruptions.

A simplified description of TCP is that it has two states of operation: *congestion avoidance* and *slow start*. A congestion window sets the number of segments (i.e., packets) that are sent each round trip time. During the congestion avoidance state, the congestion window size increases additively as long as no packets are lost. When a loss occurs, the window size is reduced multiplicatively by a factor of two. A timer that waits for a corresponding ack determines whether a packet is lost. The timeout is set as $RTO = f(EstimatedRTT, DevRTT)$, where $f()$ is a linear function of the estimated RTT and the deviation in RTT. If a loss leads to a timeout, TCP doubles the timeout value, reduces the congestion window size to 1 packet, and enters the slow start state. During slow start the window size increases exponentially until TCP enters the congestion avoidance state again.

In our model of disruptions, we let T_d be the length of disruption, and we assume that T_d is sufficiently long to generate a TCP timeout. For the Wi-Fi scenarios in which we are interested, the Internet RTT is often 100–200 ms or less; hence, our model is appropriate for disruptions of 100–200 ms or longer (assuming low variation in the RTT). When a disruption occurs and TCP times out, the congestion window size is reduced to 1, the first unacknowledged packet in the window is retransmitted, and the timeout value is doubled. If timeouts continue to occur, the timer length grows exponentially until the disruption period ends and an acknowledgment is received.

We follow the same notation as Padhey et al [22]: Let T_0 be the value of retransmission timeout, RTO, before the first timeout occurs; and let k be the number of timeouts before the disruption period ends. The sum duration of first k timeout values is given by L_k below [22]. Note that TCP fixes its maximum increase to the timeout value to $64T_0$.

$$L_k = \begin{cases} (2^k - 1)T_0 & \text{for } k \leq 6 \\ (63 + 64(k-6))T_0 & \text{for } k \geq 7 \end{cases} \quad (1)$$

The number of timeouts k for a period T_d is therefore given by

$$k = \begin{cases} \lceil log_2(\frac{T_d}{T_0} + 1) - 1 \rceil & \text{for } T_d \leq (2^6 - 1)T_0 \\ \lceil \frac{T_d + 321T_0}{64T_0} \rceil & \text{otherwise} \end{cases} \quad (2)$$

When the disruption ends, TCP does not immediately start transferring data because it has no way of knowing if a good connection to the destination is available. Instead it waits for the retransmission timer, which unfortunately increases in length exponentially during the length of the disconnection.

Specifically, after the disruption ends, the client waits for a period of time $t_w = L_k - T_d$ before it can start sending data. Moreover, TCP is in a slow start state at this time. The value of t_w is given by the following equation, partially via substitution of Eq. 2 into Eq. 1.

$$t_w = \begin{cases} (2^{\lceil log_2(\frac{T_d}{T_0}+1)-1 \rceil + 1} \cdot T_0) - T_d & \text{for } T_d \leq 63T_0 \\ (63 + 64(\lceil \frac{T_d + 321T_0}{64T_0} \rceil - 6))T_0 - T_d & \text{otherwise} \end{cases} \quad (3)$$

The time t_w is wasted by TCP. Interestingly, t_w is linear in the disruption length T_d when $T_d \leq (2^6 - 1)T_0$. Hence, if the length of a connected period is equal to the length of a disrupted period, and this sequence repeats itself, TCP would never be able to recover from the aftermath of disruptions. This scenario would lead to nearly zero throughput (see Figure 9 for experimental validation), something that is common in mobile networks where nodes move from one access point to another with disruptions in between.

4. COVERING HOLES WITH A LOW BANDWIDTH BRIDGE

The measurement study in the last section has two primary conclusions. *(i)* Coverage holes are common even in dense organic access point networks. These holes can be classified as *permanent holes* that exist due to poor AP coverage or *transient holes* that can occur due to factors such as channel characteristics, and association failures. *(ii)* These coverage holes have a severe adverse effect on performance of continuous TCP (and UDP) sessions. Hence, a large suite of applications such as web browsing, web search, instant messaging, and voice is likely to perform very poorly in such environments. For TCP we found that the performance degradation comes primarily from long timeouts and small congestion window sizes after the disruption occurs.

4.1 Patching Options

Patching coverage holes is challenging and non-trivial, although there are a number of existing proposals. For example, Vi-Fi [5] addresses transient holes by coordinating retransmissions by APs; it is unlikely that such a solution can be easily deployed in an unplanned Wi-Fi network where APs are unlikely to coordinate, and it does not address permanent holes. Similarly, client-based solutions that leverage diversity, such as FatVAP [15], can only be applied to areas where multiple access points are available and do not address permanent holes.

Another option is to avoid Wi-Fi and use very long range cellular/3G access; such networks impose a recurring cost for each device that the user carries, and are unavailable to a town or campus. For example, a municipality cannot offer free 3G access within its downtown, commercial area, nor can a University offer free 3G access on its campus; it's simply not an option. WiMax installations to cover large areas require a license, large towers, and carrier grade hardware (and will not be devoid of coverage holes).

Range	600 m
Max Data Rate	115.2 Kbps
UDP Throughput	47 Kbps
Lowest Transmit Power	1 mW
Highest Transmit Power	1 W
Receive Power	360 mW

Table 1: Characteristics of the Digi-XTend radios.

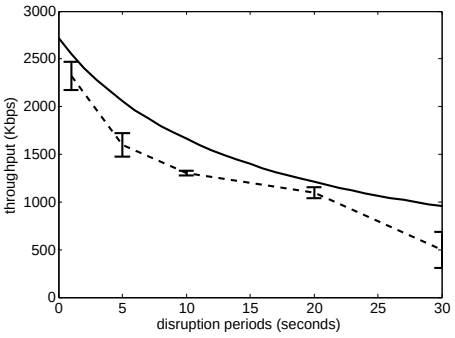

Figure 11: Validation of the analytical model for disruptions with empirical data. We find that the error between the model and experimental results is less than 20% on an average.

Finally, the most obvious way to cover WiFi holes is to add WiFi infrastructure, but that has its own challenges [7, 13]. In general, *moderately* covering a large area with medium range WiFi APs is much less expensive than *completely* covering the area so that no performance problems are present [18].

In sum, we desire an inexpensive, long-range solution — off-the-shelf, unlicensed 900 MHz radios offer both characteristics. Moreover, they have a small energy profile and can be supported by solar panels and still be portable. What such radios lack is bandwidth. A summary of the characteristics of the 900 MHz Digi-XTend radio is given in Table 1. As we demonstrate in this section, the benefits of the low-bandwidth radio bridges are well beyond their offered bandwidth.

4.2 Modeling the Benefits of Low Bandwidth Bridges

At first thought, it seems counter-intuitive to augment a WiFi network with a radio with a data rate that is $\frac{1}{20}$ as large. This approach is explained by our model in the previous section, which shows that one of the primary factors affecting TCP performance is the exponential growth of timeouts. The timeouts result in a stalled period after connections are re-established, a period that can grow linearly with the disruption length. Moreover, TCP has to restart from a window size of one (slow start) after this period.

Hence, our goal for the low-bandwidth bridge is to avoid this wasted period by keeping TCP in congestion avoidance through the period of WiFi disconnection and into a reconnection. As we show in this section, the throughput gains are substantial. Moreover, the bandwidth of 44 kbps offered

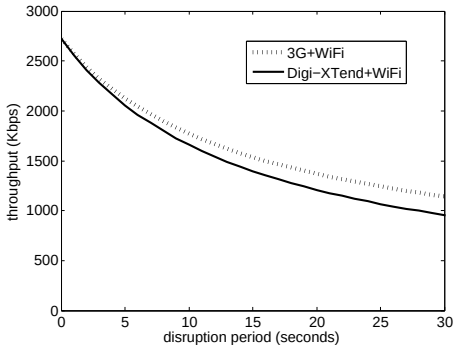

Figure 12: Comparison of using a high bandwidth radio such as 3G against using a low bandwidth Digi-XTend radio. We find that the additional benefits of the 3G's higher bandwidth is less than 14% on average.

for TCP and 47 Kbps for UDP (see Table 1) is significant, especially in the presence of larger coverage holes. VoIP codecs such as G.726 and G.728 require bit rates of less than 50 Kbps and would work well using just the Digi-XTends. To make our discussion concrete, presently we analytically demonstrate the performance benefits that a low-bandwidth augmentation can provide.

We derive a very simplified model for the aggregate send-rate with and without using a second radio. The mobile node starts on WiFi AP R_A, switches to low-bandwidth bridge R_X for the period T_d of WiFi disruption, and switches again to WiFi AP R_B when it is available. When the nodes switches from R_X to R_B, it takes time recovery time t_r before TCP reaches congestion avoidance using R_B.

We treat $t_r = 0$; this simplification is valid since, with R_X in place, the mobile node will not switch out of congestion avoidance when it begins using R_B. We assume a simple model for occurrence of coverage holes where the disrupted period is T_d followed by a period of connectivity, T_O, followed by another disruption period of T_d and so on. For the two radio system, the expected send rate is given by Equation 5. Note that B_l is the steady state send rate of the second radio and B_h is the steady-state send rate of Wi-Fi (from Equation 33 in [22]).

$$S = T_d * B_l(p_l, RTT_h) + T_O * B_h(p_h, RTT_h) \quad (4)$$

The RTT for Wi-Fi is RTT$_h$, and the RTT of the second radio is RTT$_l$. The packet loss probability for the WiFi channel is p_h and the packet loss probability for the second radio channel is p_l. For the system which uses only the WiFi radio, the average send rate of the system is the following.

$$S = T_d * B_l(p_l, RTT_h) + (T_O - t_w) * B_h(p_h, RTT_h), \quad (5)$$

where t_w is derived in Equation 3.

Figure 11 show the experimental validation of this model. The experimental line is the same as Epsilon line in Figure 16.

The model parameters, such as RTT over WiFi and the loss probability, are measured using the same setup as Figure 16. From the figure we find that the error between the model and the experiment is less than 20% on average. The deviations are because we assume steady state behavior even for small coverage holes.

Now, we use this model to compare the benefit of using a low bandwidth Digi-XTend radios and a high-bandwidth Sierra Wireless 3G modem. We measure values for p and RTT for both radios using `iperf` on UDP sessions to an Internet host. We see that the benefit of using a radio like 3G is small, less than 15% on an average. This is because the wasted time t_w is same for both 3G and Digi-XTends. Since both radios can put TCP into congestion avoidance before moving to Wi-Fi, the only gain that 3G has over Digi-XTends is the additional bandwidth provided during the disruption period. Therefore, with longer disruption periods 3G has better performance over Digi-XTends.

The other benefit of the Digi-XTend radio is that it supports several efficient cyclic sleep modes and has low-power transmit modes. The receive power of the radio is 360 mW (three times less than WiFi and 3G) while the average transmit power is around 500 mW (also three times less than Wi-Fi and 3G). We are presently working on porting our system to solar power mote class devices using XTends for nomadic deployments.

The primary decision that has to be made by the mobile node is when to switch between the two radios. Multiplexing or striping schemes for managing the single TCP stream over both radios will result in performance that is governed by the low bandwidth channel and will be affected greatly by the fluid connectivity changes. We expect the systems to be available simultaneously only at times. Epsilon, in its present implementation detects disruptions *in situ* and switches to the 900MHz radio whenever a disruption is detected. Disruption in WiFi connectivity can be detected by the absence of end-to-end connectivity (losing consecutive pings) or through lack of link-layer acknowledgments. We expect that the loss of connectivity using link-layer information can be detected quickly (on the order of 10s of milliseconds).

5. EPSILON: IMPLEMENTATION

The main components of the Epsilon architecture are shown in Figure 13. The core architecture is proxy-based, similar to the MAR system [23]. There are three types of nodes in network: (1) the mobile node router, (2) bridge nodes, (3) and an Internet proxy. In this section, we describe the design and implementation of the software modules.

5.1 Mobile node router

Each mobile node in the network is assumed to have two radios: a Wi-Fi radio and a Digi-XTend radio. Epsilon uses a user-space proxy at each mobile client. When a disruption is detected, the proxy sends packets over the low bandwidth channel; otherwise it forwards them on Wi-Fi.

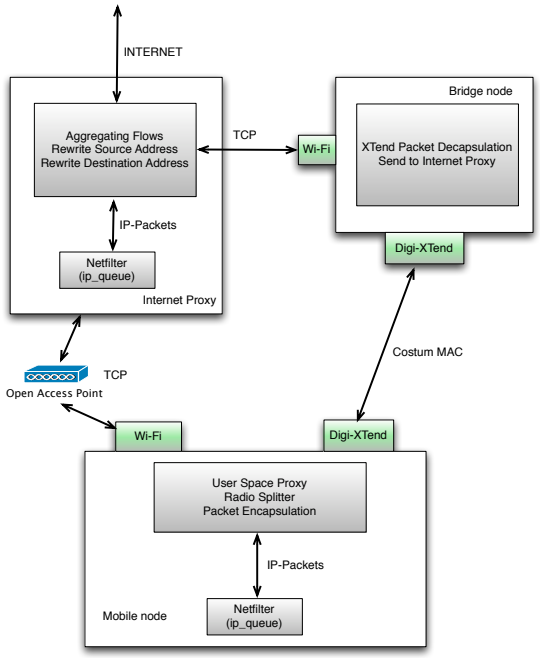

Figure 13: An overview of the Epsilon system architecture.

As the mobile node associates with different access points, the IP address associated with its outbound wireless interface changes. To mask the effect of these changes, Epsilon uses a virtual dummy interface (`eth2`) and forces all application packets to be routed through this interface. The interface is assigned a static IP address which is unique for every mobile node. However, a single static route is set to the Internet proxy through the actual outbound interface (`wlan0`). The proxy uses `Netfilter` and `ip_queue` to capture IP packets from the kernel. If the mobile node proxy decides to route packets over Wi-Fi, it encapsulates the captured IP packet in another IP packet and sends it to the Internet proxy over a TCP connection. Otherwise, the IP packet is encapsulated in a Digi-XTend packet and transferred over the 900 MHz link to the bridge.

5.2 Bridge Nodes

The bridge node is a Linux box equipped with a 900 MHz Digi-XTend radio as well as a Wi-Fi radio. It acts as an intermediary between the mobile nodes and the Internet proxy. When receiving a packet over the Digi-XTend radio, the bridge first decapsulates it to get the raw IP packet, and then encapsulates it again in another IP packet. The bridge will then forward the packet over a TCP connection to the proxy. The reverse of the process occurs when it receives a packet from the Internet proxy.

5.3 Internet proxy

The Internet proxy acts as an aggregation point for packets being received from or sent to the bridge or mobile node.

The following example demonstrates how the Internet proxy is used in Epsilon. Consider a mobile user sending a query to `google.com`. The IP packets sent from the bridge and the mobile node (encapsulated in other IP packets) have the address of the virtual interface of the mobile node as their source and `google.com` as their destination addresses. The Internet proxy rewrites the source address of the packets with its own address, recalculates the IP/TCP checksum and then sends the packets to the destination (here, `google.com`) using a raw socket. On the other hand, the packets received from `google.com` are queued using a `iptable` hook and trapped at the Internet proxy using `ip_queue`. The destination address of the packets is then rewritten with the destination address of the virtual interface of the mobile node. The packets are then sent over a previously established TCP connection with the mobile node or to the bridge from where they are sent to the mobile node over the Digi-XTends. The packets are decapsulated at the mobile node and then sent to the application using raw sockets. The mobile node reinitiates TCP connection with the Internet proxy every time it associates with a new open access point.

6. SYSTEM EVALUATION

We evaluate Epsilon by focusing on the following key questions.

1. By how much does Epsilon improve throughput of TCP sessions?
2. What performance benefits does Epsilon have for applications, such as HTTP transfers?
3. How does Epsilon impact the performance of UDP sessions that require uninterrupted connectivity?

For evaluating these factors, we use transport-layer disruption traces from our mobile testbed. Moreover, to extend our results to other testbeds, we evaluate more general scenarios using synthetic workloads.

6.1 Evaluation Methodology

We have built a prototype of the Epsilon bridge node using the same hardware deployed on our mobile nodes, described in Section 2.1. In our evaluation, we compare two systems: *Epsilon*: our proposed dual-radio and bridge system; and *Wi-Fi*: a system that uses only the Wi-Fi radio.

For our experimental evaluation, we use traces of connectivity durations collected from our testbed to carry out trace-driven emulations on an indoor experimental setup. The computer representing the mobile node begins a TCP/UDP transfers to a site on the Internet using the Wi-Fi network. Based on the traces of connectivity, the user-space proxy drops outgoing and incoming packets on the Wi-Fi interface. If the mobile node decides to use the XTend radio, it connects and transfers data to an implementation of an Epsilon bridge node, also placed in our building.

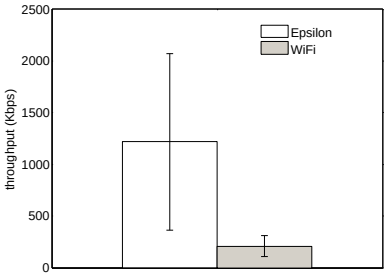

Figure 14: The TCP improvement produced by Epsilon. Epsilon produces an improvement of 6x in TCP throughput over a system which does not use the Digi-XTend radio.

While we are currently working towards deploying Epsilon on our mobile nodes, the trace-based emulations helps us understand the gains in performance that Epsilon provides for TCP and UDP sessions. Although the experimental setup is static and indoors, the effects of isolated disruptions are captured in the real world traces we have collected from our testbed in February 2009. Moreover, we saw in Section 2 that throughput is not correlated to speed, hence our results should be generally applicable to mobile environments. Accordingly, the experimentation using trace-driven emulation has two major advantages. First, it provides us the flexibility to produce repeatable results and to try out various approaches without redeployment in the testbed; and second, using a full implementation on real hardware provides a comparison platform that has accurate system delays and other key performance parameters. In our experimentation, we assume that the mobile node immediately knows when a disruption occurs, starts dropping packets on the WiFi interface and starts sending packets over the Digi-XTend radio. This is a safe assumption to make since the time within loss of WiFi connectivity can be detected is usually very small—for example, our experiments show that pinging the base station to which the node was associated can take less than 10 ms or detecting link layer retransmissions takes less than 10 ms. Given that the RTT for Wi-Fi can be 100-200 ms, assuming that detection of disruption is instantaneous is a reasonable approximation.

6.2 TCP and UDP performance

We measured TCP performance using an `iperf` server running on a remote Internet host, and an `iperf` client running on the emulated mobile node. All experiments are based on 4-minute long sessions initiated continuously during the experiment by the client's `iperf` software. The proxy on the emulated mobile node drops packets when connectivity is absent.

The results in Figure 14 show a 6x improvement in average TCP throughput for Epsilon over just WiFi, i.e., from 200 Kbps to 1200 Kbps. Note that the additional TCP throughput that the Digi-XTends can provide is limited to 50 Kbps —-hence the extra 950 Kbps is an outcome of the 900MHz radio preventing TCP timeouts (hence minimizing

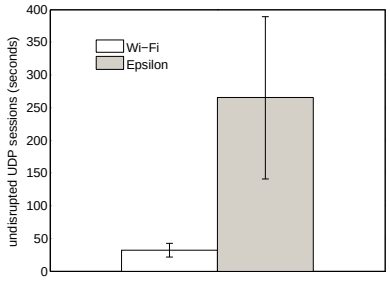

Figure 15: The length of non-disrupted UDP sessions with and without Epsilon. Epsilon increases the length of uninterrupted UDP sessions by a factor of 8.5x.

Metric	Wi-Fi	Epsilon
Web-pages downloaded	796	1187
HTTP timeouts	203	61

Table 2: Improvement produced by Epsilon for web transfers. Epsilon downloads 40% more web-pages and has less than 3x timeouts.

the wasted time t_w), prevents TCP from moving into slow start, and keeps the TCP in congestion avoidance. However, there is large variance in the Epsilon bar (shown by the error bars). Since, we run 4-minute TCP sessions and in our mobile testbed we have disruptions greater than 4 minutes, some TCP sessions fall into regimes where there is only 900MHz connectivity—gaining throughput of less than 50 Kbps.

To measure UDP performance, we modify the client `iperf` to initiate a 30-minute UDP session. The bandwidth of the connection is limited during the experiment to 500 Kbps. Figure 15 shows the length of non-disrupted periods experienced by UDP. We define the ON period as the time when the bandwidth is above 40 Kbps. Such a bandwidth is sufficient for applications such as VoIP and is close to what a GPRS connection can provide. From the results we find that Epsilon is able to increase the average length of the ON periods by a factor of 8.5x. Another result (not presented in the paper due to space limitations) show that Epsilon reduces the length of the OFF periods by a factor of 13x. These improvements are simple to explain. Epsilon, with the help of the 900MHz radio, provides near ubiquitous connectivity in the presence of disruption holes. Applications that require low bandwidth but continuous connectivity can get an order of magnitude improvement using Epsilon's infrastructure.

6.3 Application performance

Our evaluation of TCP and UDP streams show order of magnitude improvement using Epsilon. Here we capture the performance of specific applications over Epsilon. We evaluated HTTP performance since it is the basis of popular Web browsing, Web email, and Web search applications. Instant messaging, file transfers, and other applications that rely on continuous, long-running TCP streams, will achieve improvements at least as good or better than HTTP-based applications using short-lived TCP connections.

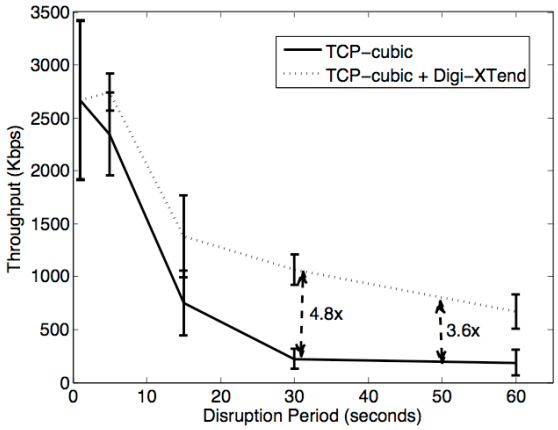

Figure 16: The figure shows TCP throughput as a function of the Off period with and without Epsilon. The disruption duration is varied and the ON period is chosen uniformly at random between 0-30 seconds.

In our web transfer experiments, we chose 10 popular web-pages and fetched their content using `wget`. The web-pages were spread across sources in U.S., Europe, and Asia. This two-hour experiment was based on two key performance metrics: the number of web-pages successfully downloaded; and the number of HTTP timeouts. The number of timeouts is proportional to the Web browsing experience a user would have. This is because timeouts lead to broken images, unloaded web-pages, and error messages.

Table 2 shows the relative benefits of using the Epsilon system for many Web application. Epsilon is able to download more than 40% extra pages and experience 3x less timeouts. While web transfers involve short TCP transfers and are probably the worst case for Epsilon as compared to Wi-Fi, we still see a substantial improvement in performance using the additional radio.

6.4 Synthetic Workloads

While these performance improvements in TCP, UDP, and application performance demonstrate Epsilon's benefits, the results are specific to our testbed. To generalize the results to other scenarios, we evaluated our system using a parametrized, synthetic connectivity trace.

For generating the synthetic trace, we simply vary two parameters. The *on* period is when connectivity is not disrupted, and the *off* period is when connectivity is disrupted. We performed the experiments using the same `iperf`-based sessions from our emulated client node to an Internet host. Figure 16 shows TCP performance as a function of the *off* period. The *on* period is chosen uniformly at random between 0–30 seconds. As the disruption periods increase the benefits of using Epsilon also increase. When the length of the *off* period is comparable to the *on* period we see a 4.6x improve-

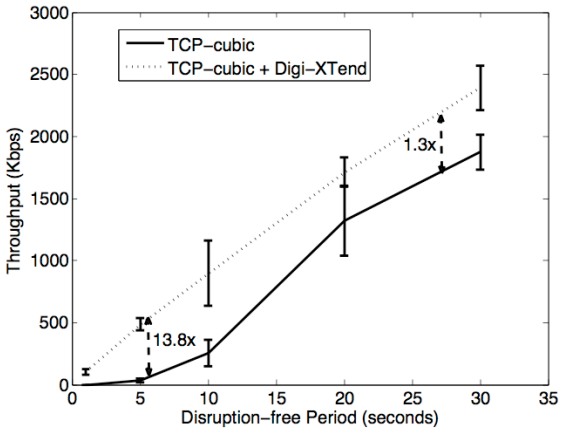

Figure 17: The figure shows TCP throughput as a function of the disruption-free/connected period with and without Epsilon. The disruption period is chosen uniformly at random between 0-30 seconds.

ment using Epsilon. Figure 17 shows the benefits of Epsilon as the *on* period varies. The right hand side of the graph represents a very dense deployment of APs while the left hand corner represents a sparse distribution of access points. Cabernet [9] reports a median duration of 4 seconds for the *on* period and an *off* period (i.e., the median time between Wi-Fi encounters) as 32 seconds. For these values, the results in Figure 16 show that Epsilon can provide a 13.8x improvement in TCP throughput. While the best known system [9] reports 2x improvement to TCP when associated with access points (effectively a 15% improvement when there are disruptions) Epsilon produces more than an order of magnitude improvement to TCP.

7. RELATED WORK

Epsilon builds on related work in the field of mobile networking, TCP for wireless, measurements of Wi-Fi connectivity from mobile nodes, and mobile network hand-off. Here we compare and contrast Epsilon with most related literature.

Measurement Studies of Wi-Fi connectivity.

Studies of Wi-Fi connectivity have been performed for application layer connectivity [11, 12, 21] and link layer connectivity [18]. Other works have focused on the performance of managed Wi-Fi access points [1, 8, 14, 17, 19]. While many studies consider simplified scenarios under controlled environments, our measurement study uses data from an unplanned Wi-Fi network. Moreover, we combine transport layer connectivity with link layer connectivity data to draw a general concrete set of conclusions on the presence and causes of coverage holes. We take a step further and design Epsilon, which uses a longer-range, low-bandwidth, inexpensive Digi-XTend radio and a simple history-based prediction algorithm to patch coverage holes.

Improving TCP for wireless .

A large body of work is related to altering TCP to perform better in a wireless lossy environment [2, 9]. These works build solutions that differentiate congestion in the wired link from losses on the wireless link. Hence, they prevent TCP from reducing its congestion window when there is a packet loss as opposed to congestion in the wired network. Epsilon is complementary to these schemes, as it addresses longer term outages rather than link layer packet loss. Other solutions propose modification of access points to improve performance of TCP/UDP in wireless networks [5, 16]. We argue that such a solution is not always feasible, given that most access points are installed by third-party individuals. Although Epsilon proposes using additional infrastructure to enhance Wi-Fi mobile network, the enhancement is an addition of cheap, easily deployable Digi-XTend radios. Other solutions such as FatVAP [15], and Juggler [20] use client side modifications to associate with multiple access points. In this work, we show that despite using beacons from multiple APs as those works suggest, coverage holes are prominent. Moreover, these solutions are limited to covering transient holes.

Hand-off and Masking disruption.

Several papers advance fast hand-offs in wireless networks [23, 25]. These solutions are plausible in a Wi-Fi network only if the length of disruptions are short. Contrary to this, in our measurement study we observe large disruptions (on the order of minutes) due to a lack of Wi-Fi coverage. Such disruptions can not be masked by using faster hand-off schemes. Epsilon provides a general solutions to patching all types of holes—small and large, transient and permanent.

Disruption tolerant networking.

A large body of work concentrates on designing delay tolerant solutions in a mobile setting [6]. While certain applications are delay tolerant such as low priority email and data collection, other applications such as web search, instant messaging, and high priority email require continuous connectivity and uninterrupted TCP sessions. Our solution would work equally well for delay tolerant and delay sensitive applications.

8. FUTURE WORK

As a part of future work, we plan to deploy solar powered Epsilon nodes which would provide connectivity for mobile nodes far from the Epsilon bridge. We have built the hardware prototype for these nodes using a tinynode mote, a custom power supply board (with current accumulator ICs to monitor energy consumption), and a Digi-XTend radio. Such a network of solar powered nodes would help patch large coverage holes. However, a major challenge to their deployment is energy management on the nodes.

The amount of energy consumed at these nodes is directly proportional to the amount of data they receive and send

over the 900MHz channel. Hence, reducing the quantity of data sent using the 900MHz radio can significantly reduce the energy burden on the nomadic nodes. While the present incarnation of Epsilon switches to the Digi-XTends whenever Wi-Fi is not available, we have analytically shown that for disruptions smaller than RTT_l (round trip time of the low bandwidth radio), it is not useful to switch to the 900MHz radio. This is because the transient time t_s, before TCP learns the correct RTT of the low bandwidth radio can be shown to be at least RTT_l if the window size at the beginning of the disruption is 1. If the window size is greater than 1 the time before TCP converges to the correct RTT is at least $2 \cdot RTT_l$. However, implementing this switching algorithm requires accurately predicting coverage holes.

We have implemented and performed some preliminary evaluation of a history based coverage holes detection algorithm which learns the duration of coverage holes in 100x100 m^2 grid. We calculated the expected period of disruptions in a region as the exponentially weighted average of the disruption lengths recently observed and the expected disruption length from history. While our prediction scheme has high accuracy of detecting holes (a median error of less than 5 seconds) in our mobile testbed, our preliminary evaluation shows that we can achieve about 18% reduction in the amount of data that can be transferred at the cost of 10% reduction in average bandwidth. The energy savings is small since in our network coverage holes are large. However, for denser networks, such as VanLan, we conjecture that our energy savings would be much higher. We plan to evaluate our algorithm on denser networks as part of future work.

9. CONCLUSION

While a great number of cities have an organic network open WiFi access points, coverage holes limit the robustness of opportunistic access by mobile users. Similarly, it is difficult to prevent holes in planned networks without very dense deployments. A measurement study of our own network shows large coverage holes, although open WiFi is available in 75% of the city. Connectivity traces from a second city show similar problems. Epsilon is our proposal to enhance existing deployments of WiFi networks, planned or organic, by placing low-bandwidth, long-range, solar-powered radios wherever holes are present, bridging a connection to near by 802.11 AP.

Our experiments show that a low bandwidth second radio has a *multiplicative* effect on the overall bandwidth for the mobile user. Our use of the Digi-XTend 115 kpbs radio to covering 802.11 holes can increase the aggregate TCP throughput by 1.2–13x, well beyond the data it carries. This large benefit is due to the ability of the low-bandwidth channel to keep the TCP sender's window large despite connectivity problems. Further, this enhancement requires no modifications to TCP, making it easy to deploy in existing systems.

10. REFERENCES

[1] M. Afanasyev, T. Chen, G. M. Voelker, and A. C. Snoeren. Analysis of a Mixed-Use Urban WiFi Network: When Metropolitan becomes Neapolitan. In *Proceedings of IMC*, October 2008.

[2] H. Balakrishnan, V. Padmanabhan, and R. Katz. The Effects of Asymmetry on TCP Performance. *ACM Mobile Networks and Applications (MONET)*, 1999.

[3] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. In *SIGCOMM '96*, pages 256–269, 1996.

[4] A. Balasubramanian, B. N. Levine, and A. Venkataramani. Enabling Interactive Applications in Hybrid Networks. In *Proc. ACM Mobicom*, September 2008.

[5] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. Levine, and J. Zahorjan. Interactive WiFi Connectivity for Moving Vehicles. In *Proc. ACM SIGCOMM*, August 2008.

[6] A. Balasubramanian, Y. Zhou, W. B. Croft, B. N. Levine, and A. Venkataramani. Web Search From a Bus. In *Proc. ACM Workshop on Challenged Networks (CHANTS)*, September 2007.

[7] N. Banerjee, M. D. Corner, D. Towsley, and B. N. Levine. Relays, Base Stations, and Meshes: Enhancing Mobile Networks with Infrastructure. In *Proceedings of ACM MobiCom*, San Francisco, CA, USA, September 2008.

[8] J. Camp, J. Robinson, C. Steger, and E. Knightly. Measurement driven deployment of a two-tier urban mesh access network. In *Proc. ACM MobiSys*, pages 96–109, 2006.

[9] J. Eriksson, H. Balakrishnan, and S. Madden. Cabernet: Vehicular Content Delivery Using WiFi. In *Proceedings of ACM MobiCom*, San Francisco, CA, September 2008.

[10] K. Fall. A Delay-Tolerant Network Architecture for Challenged Internets. In *Proc. ACM Sigcomm*, pages 27–34, 2003.

[11] R. Gass, J. Scott, and C. Diot. Measurements of In-Motion 802.11 Networking. In *WMCSA '06: Proc. of the Seventh IEEE Workshop on Mobile Computing Systems & Applications*, pages 69–74, 2006.

[12] D. J. Goodman, J. Borras, N. B. Mandayam, and R. D. Yates. INFOSTATIONS: A new system model for data and messaging services. In *Proc. Vehicular Technology Conference*, pages 969–973, Phoenix, AZ, May 1997.

[13] P. Hui, A. Lindgren, and J. Crowcroft. Empirical Evaluation of Hybrid Opportunistic Networks. In *Proceedings of COMSNETS*, Bangalore, India, January 2009.

[14] S. B. John Bicket, Daniel Aguayo and A. . Robert Morris, Mobicom 2005. Architecture and evaluation of an unplanned 802.11b mesh network. In *Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 31–42, August 2005.

[15] S. Kandula, K. C.-J. Lin, T. Badirkhanli, and D. Katabi. FatVAP: Aggregating AP Backhaul Capacity to Maximize Throughput. In *5th USENIX Symposium on Networked Systems Design and Implementation*, San Francisco, CA, April 2008.

[16] U. Kubach and K. Rothermel. Exploiting Location Information for Infostation-based Hoarding. In *In Proc. of MOBICOM 2001*, pages 217–230, 2004.

[17] H. Lundgren, K. Ramach, E. Belding-royer, K. Almeroth, M. Benny, A. Hewatt, E. Touma, and A. Jardosh. Experiences from the design, deployment, and usage of the ucsb meshnet testbed. *IEEE Wireless Communications*, 13(2):18–29, April 2006.

[18] R. Mahajan, J. Zahorjan, and B. Zill. Understanding WiFi-based Connectivity From Moving Vehicles. In *Proc. Internet Measurement Conference (IMC)*, 2007.

[19] R. Murty, G. Mainland, I. Rose, A. R. Chowdhury, A. Gosain, J. Bers, and M. Welsh. Citysense: A vision for an urban-scale wireless networking testbed. In *Proceedings of the IEEE International Conference on Technologies for Homeland Security*, May 2008.

[20] A. J. Nicholson, S. Wolchok, , and B. D. Noble. Juggler: Virtual Networks for Fun and Profit. Technical Report CSE-TR-542-08, University of Michigan, April 2008.

[21] J. Ott and D. Kutscher. Drive-Thru Internet: IEEE 802.11b for "Automobile" Users. In *Proc. IEEE INFOCOM*, pages 362–373, March 2004.

[22] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation . In *ACM Sigcomm*, 1998.

[23] P. Rodriguez, I. Pratt, R. Chakravorty, and S. Banerjee. Mar: A commuter router infrastructure for the mobile internet. In *In Proc. of ACM Mobisys*, pages 217–230, 2004.

[24] N. Sastry, J. Crowcroft, and K. Sollins. Architecting Citywide Ubiquitous Wi-Fi Access. In *Proceedings of HotNets 2007*, Atlanta, Georgia, November 2007.

[25] M. H. Shin, A. Mishra, and W. Arbaugh. Improving the Latency of 802.11 hand-offs using neighbor graphs. In *Proc. of Mobisys*, pages 69–74, 2004.